

Necklaces, Convolutions, and $X + Y$

| | | |
|-------------------------------|------------------------------|-------------------------------|
| David Bremner ¹ | Timothy M. Chan ² | Erik D. Demaine ³ |
| Jeff Erickson ⁴ | Ferran Hurtado ⁵ | John Iacono ⁶ |
| Stefan Langerman ⁷ | Mihai Pătraşcu ⁸ | Perouz Taslakian ⁹ |

In memory of our colleague Mihai Pătraşcu.

Abstract

We give subquadratic algorithms that, given two necklaces each with n beads at arbitrary positions, compute the optimal rotation of the necklaces to best align the beads. Here alignment is measured according to the ℓ_p norm of the vector of distances between pairs of beads from opposite necklaces in the best perfect matching. We show surprisingly different results for $p = 1$, p even, and $p = \infty$. For p even, we reduce the problem to standard convolution, while for $p = \infty$ and $p = 1$, we reduce the problem to $(\min, +)$ convolution and $(\text{median}, +)$ convolution. Then we solve the latter two convolution problems in subquadratic time, which are interesting results in their own right. These results shed some light on the classic sorting $X + Y$ problem, because the convolutions can be viewed as computing order statistics on the antidiagonals of the $X + Y$ matrix. All of our algorithms run in $o(n^2)$ time, whereas the obvious algorithms for these problems run in $\Theta(n^2)$ time.

¹Faculty of Computer Science, University of New Brunswick, Fredericton, New Brunswick, Canada, bremner@unb.ca. Supported by NSERC.

²School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, tmchan@uwaterloo.ca. Supported by NSERC.

³Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, edemaine@mit.edu. Supported in part by NSF grants CCF-0430849 and OISE-0334653 and by an Alfred P. Sloan Fellowship.

⁴Computer Science Department, University of Illinois, Urbana-Champaign, IL, USA, jeffe@cs.uiuc.edu.

⁵Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, Spain, Ferran.Hurtado@upc.edu. Supported in part by projects MICINN MTM2009-07242, Gen. Cat. DGR 2009SGR1040, and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: MICINN Project EUI-EURC-2011-4306, for Spain.

⁶Department of Computer and Information Science, Polytechnic University, Brooklyn, NY, USA, http://john.poly.edu. Supported in part by NSF grants CCF-0430849 and OISE-0334653 and by an Alfred P. Sloan Fellowship.

⁷Directeur de Recherches du FRS-FNRS, Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium, stefan.langerman@ulb.ac.be.

⁸Chercheur qualifié du FNRS, Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium, stefan.langerman@ulb.ac.be.

⁹College of Science and Engineering, American University of Armenia, Yerevan, Armenia, ptaslakian@aua.am.

1 Introduction

How should we rotate two necklaces, each with n beads at different locations, to best align the beads? More precisely, each necklace is represented by a set of n points on the unit-circumference circle, and the goal is to find rotations of the necklaces, and a perfect matching between the beads of the two necklaces, that minimizes some norm of the circular distances between matched beads. In particular, the ℓ_1 norm minimizes the average absolute circular distance between matched beads, the ℓ_2 norm minimizes the average squared circular distance between matched beads, and the ℓ_∞ norm minimizes the maximum circular distance between matched beads. The ℓ_1 version of this necklace alignment problem was introduced by Toussaint [39] in the context of comparing rhythms in computational music theory, with possible applications to rhythm phylogeny [22, 40].

Toussaint [39] gave a simple $O(n^2)$ -time algorithm for ℓ_1 necklace alignment, and highlighted as an interesting open question whether the problem could be solved in $o(n^2)$ time. In this paper, we solve this open problem by giving $o(n^2)$ -time algorithms for ℓ_1 , ℓ_2 , and ℓ_∞ necklace alignment, in both the standard real RAM model of computation and the less realistic nonuniform linear decision tree model of computation. Our results for the case of the ℓ_1 and ℓ_∞ distance measures in the real RAM model also answer the questions posed by Clifford et al. in [13] (see the *shift matching problem* in Problem 5 of the tech report).

Necklace alignment problem. More formally, in the *necklace alignment problem*, the input is a number p representing the ℓ_p norm, and two sorted vectors of n real numbers, $\vec{x} = \langle x_0, x_1, \dots, x_{n-1} \rangle$ and $\vec{y} = \langle y_0, y_1, \dots, y_{n-1} \rangle$, representing the two necklaces. See Figure 1. Canonically, we assume that each number x_i and y_i is in the range $[0, 1)$, representing a point on the unit-circumference circle (parameterized clockwise from some fixed point). The distance between two beads x_i and y_j is the minimum between the clockwise and counterclockwise distances along the circumference of the unit-perimeter circular necklaces. We define this distance as:

$$d^o(x_i, y_j) = \min\{|x_i - y_j|, (1 - |x_i - y_j|)\}.$$

The optimization problem involves two parameters. The first parameter, the *offset* $c \in [0, 1)$, is the clockwise rotation angle of the first necklace relative to the second necklace. The second parameter, the *shift* $s \in \{0, 1, \dots, n\}$, defines the perfect matching between beads: bead i of the first necklace matches with bead $(i + s) \bmod n$ of the second necklace. (Here we use the property that an optimal perfect matching between the beads does not cross itself.)

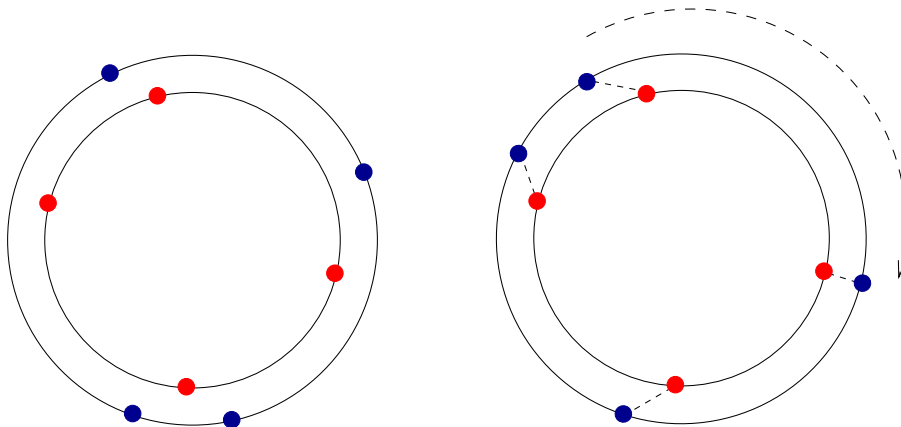


Figure 1: An example of necklace alignment: the input (left) and one possible output (right).

The goal of the ℓ_p necklace alignment problem is to find the offset $c \in [0, 1)$ and the shift $s \in \{0, 1, \dots, n\}$ that minimize

$$\sum_{i=0}^{n-1} (d^\circ((x_i + c) \bmod 1, y_{(i+s) \bmod n}))^p \quad (1)$$

or, in the case $p = \infty$, that minimize

$$\max_{i=0}^{n-1} \{d^\circ((x_i + c) \bmod 1, y_{(i+s) \bmod n})\}.$$

The ℓ_1 , ℓ_2 , and ℓ_∞ necklace alignment problems all have trivial $O(n^2)$ solutions, although this might not be obvious from the definition. In each case, as we show, the optimal offset c can be computed in linear time for a given shift value s (sometimes even independent of s). The optimization problem is thus effectively over just $s \in \{0, 1, \dots, n\}$, and the objective costs $O(n)$ time to compute for each s , giving an $O(n^2)$ -time algorithm.

Although necklaces are studied throughout mathematics, mainly in combinatorial settings, we are not aware of any work on the necklace alignment problem before Toussaint [39]. He introduced ℓ_1 necklace alignment, calling it the *cyclic swap-distance* or *necklace swap-distance* problem, with a restriction that the beads lie at integer coordinates. Ardila et al. [2] give a $O(k^2)$ -time algorithm for computing the necklace swap-distance between two binary strings, with k being the number of 1-bits (beads at integer coordinates). Colannino et al. [16] consider some different distance measures between two sets of points on the real line in which the matching does not have to match every point. They do not, however, consider alignment under such distance measures.

Aloupis et al. [1], consider the problem of computing the similarity of two melodies represented as closed orthogonal chains on a cylinder. Their goal is to find the proper (rigid) translation of one of the chains in the vertical (representing pitch) and tangential (representing time) direction so that the area between the chains is minimized. The authors present an $O(mn \lg(n + m))$ algorithm that solves the problem. When the melodic chains each have a note at every time unit, the melodic similarity problem is equivalent to the necklace alignment problem, and as our results are subquadratic, we improve on the results of Aloupis et al. [1] for this special case.

Convolution. Our approach in solving the necklace alignment problem is based on reducing it to another important problem, convolution, for which we also obtain improved algorithms. The $(+, \cdot)$ convolution of two vectors $\vec{x} = \langle x_0, x_1, \dots, x_{n-1} \rangle$ and $\vec{y} = \langle y_0, y_1, \dots, y_{n-1} \rangle$, is the vector $\vec{x} * \vec{y} = \langle z_0, z_1, \dots, z_{n-1} \rangle$ where $z_k = \sum_{i=0}^k x_i \cdot y_{k-i}$. One can generalize convolution to any (\oplus, \odot) operators. Algorithmically, a convolution with specified addition and multiplication operators (here denoted $\vec{x} \overset{\odot}{*} \vec{y}$) can be easily computed in $O(n^2)$ time. However, the $(+, \cdot)$ convolution can be computed in $O(n \lg n)$ time using the Fast Fourier Transform [18, 29, 30], because the Fourier transform converts convolution into elementwise multiplication. Indeed, fast $(+, \cdot)$ convolution was one of the early breakthroughs in algorithms, with applications to polynomial and integer multiplication [5], batch polynomial evaluation [19, Problem 30-5], 3SUM [3, 23], string matching [12, 17, 25, 31, 32], matrix multiplication [15], and even juggling [8].

In this paper we use three types of convolutions: $(\min, +)$ convolution, whose k th entry $z_k = \min_{i=0}^k \{x_i + y_{k-i}\}$; $(\text{median}, +)$ convolution, whose k th entry $z_k = \text{median}_{i=0}^k \{x_i + y_{k-i}\}$; and $(+, \cdot)$ convolution, whose k th entry $z_k = \sum_{i=0}^k \{x_i \cdot y_{k-i}\}$. As we show in Theorems 3, 6, and 17, respectively, ℓ_2 necklace alignment reduces to standard $(+, \cdot)$ convolution, ℓ_∞ necklace alignment

reduces to $(\min, +)$ [and $(\max, +)$] convolution, and ℓ_1 necklace alignment reduces to $(\text{median}, +)$ convolution. The $(\min, +)$ convolution problem has appeared frequently in the literature, already appearing in Bellman’s early work on dynamic programming in the early 1960s [4, 24, 33–35, 38]. Its name varies among “minimum convolution”, “min-sum convolution”, “inf-convolution”, “infimal convolution”, and “epigraphical sum”.¹ To date, however, no worst-case $o(n^2)$ -time algorithms for this convolution, or the more complex $(\text{median}, +)$ convolution, has been obtained (it should be noted here that the quadratic worst-case running time for $(\text{median}, +)$ convolution follows from linear-time median finding [6, 36]). In this paper, we develop worst-case $o(n^2)$ -time algorithms for $(\min, +)$ and $(\text{median}, +)$ convolution, in the real RAM and the nonuniform linear decision tree models of computation.

The only subquadratic results for $(\min, +)$ convolution concern two special cases. First, the $(\min, +)$ convolution of two convex sequences or functions can be trivially computed in $O(n)$ time by a simple merge, which is the same as computing the Minkowski sum of two convex polygons [35]. This special case is already used in image processing and computer vision [24, 33]. Second, Bussieck et al. [7] proved that the $(\min, +)$ convolution of two *randomly permuted* sequences can be computed in $O(n \lg n)$ expected time. Our results are the first to improve the worst-case running time for $(\min, +)$ convolution.

Connections to $X + Y$. The necklace alignment problems, and their corresponding convolution problems, are also intrinsically connected to problems on $X + Y$ matrices. Given two lists of n numbers, $X = \langle x_0, x_1, \dots, x_{n-1} \rangle$ and $Y = \langle y_0, y_1, \dots, y_{n-1} \rangle$, $X + Y$ is the matrix of all pairwise sums, whose (i, j) th entry is $x_i + y_j$. A classic unsolved problem [20] is whether the entries of $X + Y$ can be sorted in $o(n^2 \lg n)$ time. Fredman [27] showed that $O(n^2)$ comparisons suffice in the nonuniform linear decision tree model, but it remains open whether this can be converted into an $O(n^2)$ -time algorithm in the real RAM model. Steiger and Streinu [37] gave a simple algorithm that takes $O(n^2 \lg n)$ time while using only $O(n^2)$ comparisons.

The $(\min, +)$ convolution is equivalent to finding the minimum element in each antidiagonal of the $X + Y$ matrix, and similarly the $(\max, +)$ convolution finds the maximum element in each antidiagonal. We show that ℓ_∞ necklace alignment is equivalent to finding the antidiagonal of $X + Y$ with the smallest *range* (the maximum element minus the minimum element). The $(\text{median}, +)$ convolution is equivalent to finding the median element in each antidiagonal of the $X + Y$ matrix. We show that ℓ_1 necklace alignment is equivalent to finding the antidiagonal of $X + Y$ with the smallest *median cost* (the total distance between each element and the median of the elements). Given the apparent difficulty in sorting $X + Y$, it seems natural to believe that the minimum, maximum, and median elements of every antidiagonal cannot be found, and that the corresponding objectives cannot be minimized, any faster than $O(n^2)$ total time. Figure 2 shows a sample $X + Y$ matrix with the maximum element in each antidiagonal marked, with no apparent structure. Nonetheless, we show that $o(n^2)$ algorithms are possible.

Our results. In the standard real RAM model, we give subquadratic algorithms for the ℓ_1 , ℓ_2 , and ℓ_∞ necklace alignment problems, and for the $(\min, +)$ and $(\text{median}, +)$ convolution problems. We present:

1. an $O(n \lg n)$ -time algorithm on the real RAM for ℓ_2 necklace alignment (Section 3).

¹“Tropical convolution” would also make sense, by direct analogy with tropical geometry, but we have never seen this terminology used in print.

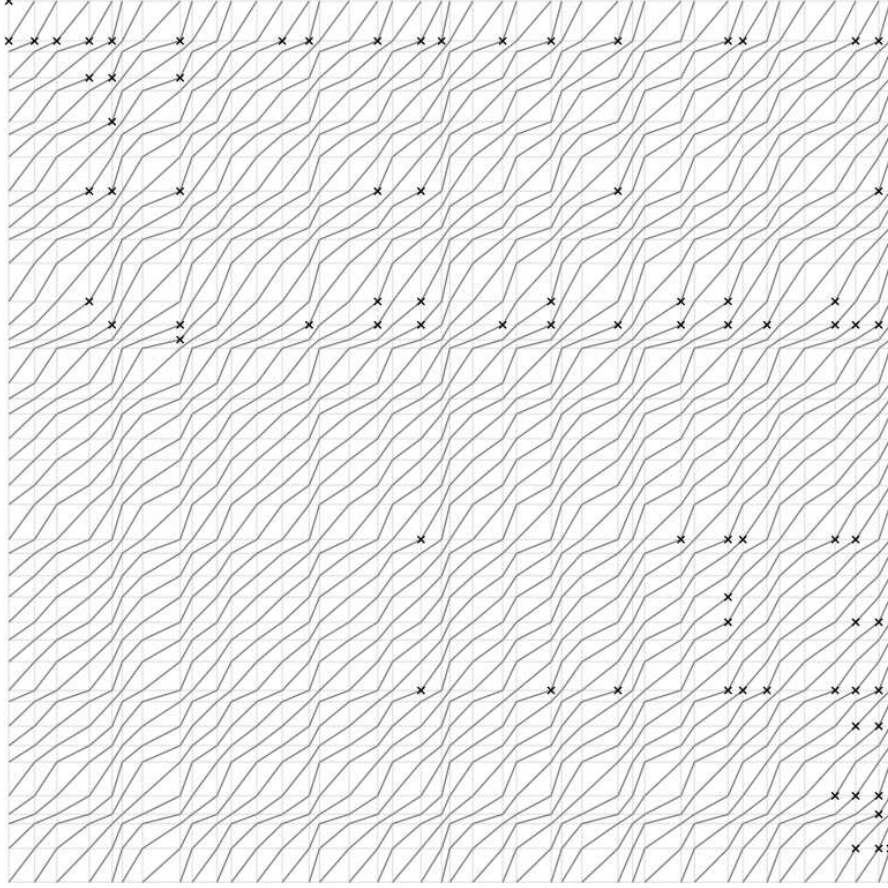


Figure 2: An $X + Y$ matrix. Each polygonal line denotes an antidiagonal of the matrix, with a point at coordinates (x, y) denoting the value $x + y$ for $x \in X$ and $y \in Y$. An \times denotes the maximum element in each antidiagonal.

2. an $O(n^2/\lg n)$ -time algorithm on the real RAM for ℓ_∞ necklace alignment and $(\min, +)$ convolution (Section 4). This algorithm uses a technique of Chan originally developed for the all-pairs shortest paths problem [9]. Despite the roughly logarithmic factor improvements for ℓ_1 and ℓ_∞ , this result does not use word-level bit tricks of word-RAM fame.
3. a further improved $O(n^2(\lg \lg n)^3/\lg^2 n)$ -time algorithm for ℓ_∞ necklace alignment and $(\min, +)$ convolution (Section 4). We actually give a direct black-box reduction of $(\min, +)$ convolution to all-pairs shortest paths; the result then follows from the current best upper bound for all-pairs shortest paths [10]. The all-pairs shortest paths works in the real RAM with respect to the inputs, i.e. it does not use bit tricks on the inputs. The algorithm, however, requires bit tricks on other numbers, but works in a standard model that assumes $(\lg n)$ -bit words.
4. an $O(n^2(\lg \lg n)^2/\lg n)$ -time algorithm on the real RAM for ℓ_1 necklace alignment and $(\text{median}, +)$ convolution (Section 5). This algorithm uses an extension of the technique of Chan [9].

In the nonuniform linear decision tree model, we give particularly fast algorithms for the ℓ_1 and ℓ_∞ necklace alignment problems, using techniques of Fredman [27, 28]:

5. $O(n\sqrt{n})$ -time algorithm in the nonuniform linear decision tree model for ℓ_∞ necklace alignment and $(\min, +)$ convolution (Section 4).

6. $O(n\sqrt{n \lg n})$ -time algorithm in the nonuniform linear decision tree model for ℓ_1 necklace alignment and (median, +) convolution (Section 5).

(Although we state our results here in terms of (min, +) and (median, +) convolution, the results below use $-$ instead of $+$ because of the synergy with necklace alignment.) We also mention connections to the venerable $X + Y$ and 3SUM problems in Section 6.

2 Linear Versus Circular Alignment

Before we proceed with proving our results, we first show that any optimal solution to the necklace alignment problem can be transformed into an optimal solution to the problem of linear alignment—aligning and matching beads that are on a line. We then can use the simpler optimization function of the “linear alignment problem” to show our results. Let $d^-(x_i, y_j) = |x_i - y_j|$ be the linear distance between two beads x_i and y_j . In the *linear alignment problem* we are given two sorted vectors of real numbers $\vec{x} = \langle x_0, x_1, \dots, x_{n-1} \rangle$ and $\vec{y} = \langle y_0, y_1, \dots, y_{m-1} \rangle$ with $m \geq n$, and we want to find s ($s < m - n$) and c that minimize

$$\sum_{i=0}^{n-1} (d^-(x_i + c, y_{i+s}))^p \quad (2)$$

or, in the case $p = \infty$, that minimize

$$\max_{i=0}^{n-1} \{d^-(x_i + c, y_{i+s})\}.$$

The main difference between (1) and (2) is that instead of taking the minimum between the clockwise and counterclockwise distances between pairs of matched beads in (1), we are simply summing the forward distances between beads in (2). We will now show that whether the beads are on a line (repeating \vec{y} infinitely many times on the line) or a circle, the optimal alignment of the beads \vec{x} and \vec{y} in these two cases are equal.

Let M° and M^- be an alignment/matching of the beads of \vec{x} and \vec{y} along the unit circumference circle C and the infinite line segment L respectively. An *edge* (x_i, y_j) of M° (M^-) is the shortest segment that connects two matched beads x_i and y_j in M° (M^-); thus, the length of (x_i, y_j) is equal to $d^\circ(x_i, y_j)$ ($d^-(x_i, y_j)$). We will show that the sum of the lengths of the edges of each of the optimal matchings $M^{\circ*}$ and M^{-*} are equal. Note that by the quadrangle inequality, we have that the edges of both $M^{\circ*}$ and M^{-*} are non-crossing.

Observation 1. *Consider any edge (x_i, y_j) along the circular necklace. If this edge crosses point 0, then the distance*

$$d^\circ(x_i, y_j) = (1 - |x_i - y_j|);$$

otherwise,

$$d^\circ(x_i, y_j) = |x_i - y_j|.$$

Let $y\vec{y}$ be the doubling of the vector \vec{y} such that

$$y\vec{y} = \langle y_0, \dots, y_{m-1}, y_0, \dots, y_{m-1} \rangle = \langle yy_0, \dots, yy_{m-1}, yy_m, \dots, yy_{2m-1} \rangle.$$

Theorem 2. *If $M^{\circ*}$ is the optimal matching of two given vectors \vec{x} and \vec{y} (both of length n) along the unit-circumference circle C and M^{-*} is the optimal matching of \vec{x} and $y\vec{y}$ along line L , then $|M^{\circ*}| = |M^{-*}|$.*

Proof. First we show that the value of any optimal matching of a set of beads along L is at least as large as the value of the optimal solution of the beads along C . Given an optimal matching $M^{-*} = \{s, c\}$ beads along L , we will “wrap” the line L around a unit circle C by mapping each of the n matched beads x_i (yy_{i+s}) along L to x_i° ($y_{(i+s) \bmod n}^\circ$) along C . (Thus we have exactly n pairs of beads along C .) Now, for every $i = 0, 1, \dots, n-1$, the length of an edge of M^{-*} is equal to

$$\begin{aligned}
& d^-(x_i + c, yy_{i+s}) \\
&= d^-(x_i + c, y_{(i+s) \bmod n}) \\
&= |x_i + c - y_{(i+s) \bmod n}| \\
&\geq \min\{|(x_i + c) \bmod 1 - y_{(i+s) \bmod n}|, (1 - |(x_i + c) \bmod 1 - y_{(i+s) \bmod n}|)\} \\
&= d^\circ((x_i^\circ + c) \bmod 1, y_{(i+s) \bmod n}^\circ).
\end{aligned}$$

Thus, as every edge length of the matching M^{-*} is at least as large as its corresponding edge along the circle C , we have $|M^{-*}| \geq |M^{\circ*}|$.

Next we show that the value of any optimal matching of a set of beads along C is at least as large as the value of the optimal solution of the beads along L . Suppose we have an optimal matching $M^{\circ*} = (s, c)$. We map every point $x_i + c$ and y_i to the infinite line segment so that the edges of $M^{\circ*}$ are preserved in M^- . Thus, for all $i = 0, 1, \dots, n-1$ and $k \in \mathbb{Z}$, we map

$$\begin{aligned}
(x_i + c) \bmod 1 &\mapsto x_i^- = x_i + c, \\
y_i &\mapsto y_{i+kn}^- = y_i + k.
\end{aligned}$$

With this transformation, in any valid matching M^- , the matched beads span at most two consecutive intervals $[k, k+1)$ and $[k+1, k+2)$ for any $k \in \mathbb{Z}$. In particular, the beads $x_i + c$ span the intervals $[0, 1)$ and $[1, 2)$.

Now we construct M^- given $M^{\circ*}$ by matching every x_i^- to y_{i+s+kn}^- such that, whenever $x_i + c < 1$ (see Figure 3),

- $k = -1$
if $((x_i + c) \bmod 1, y_{(i+s) \bmod n})$ crosses point 0 and $(x_i + c) \bmod 1 < y_{(i+s) \bmod n}$;
- $k = 1$
if $((x_i + c) \bmod 1, y_{(i+s) \bmod n})$ crosses point 0 and $(x_i + c) \bmod 1 > y_{(i+s) \bmod n}$;
- $k = 0$
if $((x_i + c) \bmod 1, y_{(i+s) \bmod n})$ does not cross point 0;

and, whenever $x_i + c \geq 1$, we increment k by 1 in each of the cases. Thus, when $x_i + c \geq 1$,

- $k = 0$
if $((x_i + c) \bmod 1, y_{(i+s) \bmod n})$ crosses point 0 and $(x_i + c) \bmod 1 < y_{(i+s) \bmod n}$;
- $k = 2$
if $((x_i + c) \bmod 1, y_{(i+s) \bmod n})$ crosses point 0 and $(x_i + c) \bmod 1 > y_{(i+s) \bmod n}$;
- $k = 1$
if $((x_i + c) \bmod 1, y_{(i+s) \bmod n})$ does not cross point 0.

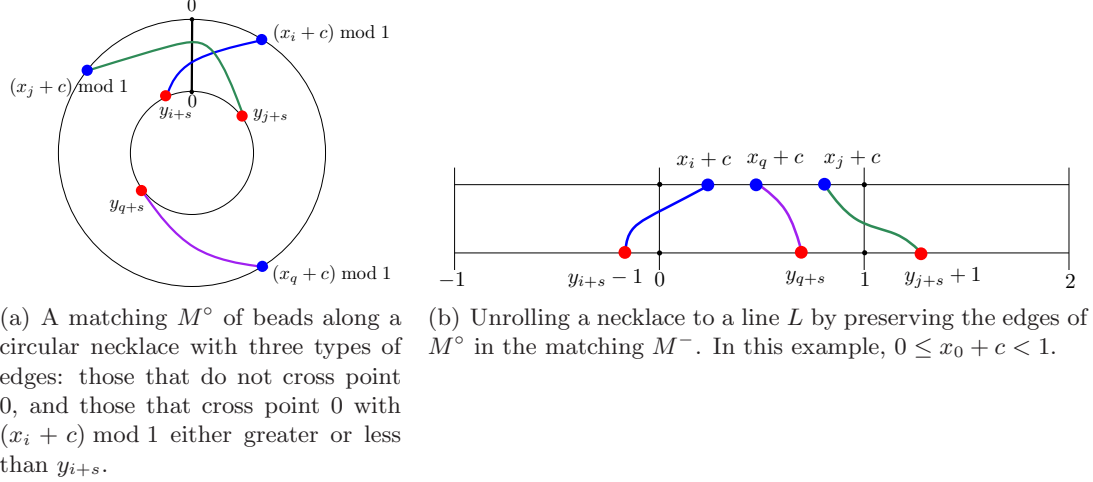


Figure 3: Unrolling a circular necklace to a line.

Here, the variable k basically decides the interval $[-1, 0)$, $[0, 1)$, or $[1, 2)$ in which the bead $y_{(i+s) \bmod n}$ is located, based on the type of the edge $((x_i + c) \bmod 1, y_{(i+s) \bmod n})$. Observe that, if an edge in $M^{\circ*}$ crosses point 0, then its corresponding edge in M^- crosses (r, r) for some $r \in \{0, 1, 2\}$.

Now, the sum of the distances of the matched beads of M^- is equal to

$$\begin{aligned} & d^-(x_0 + c, y_{(0+s) \bmod n} + k) + d^-(x_1 + c, y_{(1+s) \bmod n} + k) + \cdots + \\ & d^-(x_{n-s-1} + c, y_{n-1} + k) + d^-(x_{n-s} + c, y_0 + k + 1) + \cdots + \\ & d^-(x_{n-s-1} + c, y_{n-1} + k + 1). \end{aligned}$$

We claim that the value of $M^{\circ*}$ is equal to at least the value of this matching M^- . We show this claim by comparing the length of each edge $((x_i + c) \bmod 1, y_{i+s})$ of $M^{\circ*}$ with its corresponding edge in M^- .

Edges that do not cross point 0: If an edge of $M^{\circ*}$ does not cross point 0, then the corresponding edge in M^- does not cross any of the edges $(0, 0)$, $(1, 1)$ or $(2, 2)$; hence both endpoints (beads) of the given matching edge are within the same interval $[r, r + 1)$ for some $r \in \{0, 1\}$ (the purple edge $(x_q + c, y_{q+s})$ in Figure 3). This means that, when $x_i + c = (x_i + c) \bmod 1$, we have $k = 0$ and

$$\begin{aligned} d^-(x_i + c, y_{(i+s) \bmod n} + k) &= |(x_i + c) - (y_{(i+s) \bmod n} + k)| \\ &= |(x_i + c) \bmod 1 - (y_{(i+s) \bmod n})| \\ &= |x_i^- - y_{(i+s) \bmod n}^-| \\ &= d^\circ(x_i^-, y_{(i+s) \bmod n}^-) \quad (\text{by Observation 1}). \end{aligned}$$

We can similarly show that, when $x_i + c = (x_i + c) \bmod 1 + 1$, we have $k = 1$ and the edges of $M^{\circ*}$ that do not cross point 0 have the same length as their corresponding edge in M^- .

Edges that cross point 0: If an edge of $M^{\circ*}$ crosses point 0, then the corresponding edge in M^- crosses edge (r, r) and hence the two endpoints (beads) of the given edge of M^- must be in different and consecutive intervals: $[r - 1, r)$ and $[r, r + 1)$ for some $r \in \{0, 1, 2\}$ (the green and blue edges $(x_i + c, y_{i+s}), (x_j + c, y_{j+s})$ in Figure 3). Then, assuming $x_i + c = (x_i + c) \bmod 1$, we have

- When $(x_i + c) \bmod 1 < y_{(i+s) \bmod n}$, $k = -1$ and

$$\begin{aligned}
d^-(x_i + c, y_{(i+s) \bmod n} + k) &= |(x_i + c) - (y_{(i+s) \bmod n} - 1)| \\
&= |(x_i + c) \bmod 1 - y_{(i+s) \bmod n} + 1| \\
&= (1 - |((x_i + c) \bmod 1) - y_{(i+s) \bmod n}|) \\
&= (1 - |x_i^- - y_{((i+s) \bmod n) - n}^-|) \\
&= d^o(x_i^-, y_{((i+s) \bmod n) - n}^-) \text{ (by Observation 1).}
\end{aligned}$$

- When $(x_i + c) \bmod 1 > y_{(i+s) \bmod n}$, $k = 1$ and

$$\begin{aligned}
d^-(x_i + c, y_{(i+s) \bmod n} + k) &= |(x_i + c) - (y_{(i+s) \bmod n} + 1)| \\
&= |(x_i + c) \bmod 1 - y_{(i+s) \bmod n} - 1| \\
&= (1 - |((x_i + c) \bmod 1) - y_{(i+s) \bmod n}|) \\
&= (1 - |x_i^- - y_{((i+s) \bmod n) + n}^-|) \\
&= d^o(x_i^- + c, y_{((i+s) \bmod n) + n}^-) \text{ (by Observation 1).}
\end{aligned}$$

We can similarly show that, when $x_i + c = (x_i + c) \bmod 1 + 1$, the edges of $M^{\circ*}$ that cross point 0 have the same length as their corresponding edge in M^- .

Therefore, the length of every edge of $M^{\circ*}$ along the circle is equal to the length of its corresponding edge in M^- . Thus, the value of the matching $M^{\circ*}$ is at least as large as that of M^{-*} , completing the proof of the theorem. \square

We now proceed to prove our results by using the objective function (2).

3 ℓ_2 Necklace Alignment and $(+, \cdot)$ Convolution

In this section, we first show how ℓ_2 necklace alignment reduces to standard convolution, leading to an $O(n \lg n)$ -time algorithm that uses the Fast Fourier Transform. We then show how this result generalizes to ℓ_p for any even p . It should be noted here that the ℓ_2 necklace alignment problem was solved independently by Clifford et al. [13] (see Problem 5) using Fast Fourier Transforms. Our proof uses essentially the same technique of expanding the squared term and then optimizing terms separately, but goes through the steps in more detail; we include our proof for completeness. More results that use the FFT to solve different flavors of matching problems may also be found in [14] and [11].

Theorem 3. *The ℓ_2 necklace alignment problem can be solved in $O(n \lg n)$ time on a real RAM.*

Proof. The objective (2) expands algebraically to

$$\begin{aligned}
&\sum_{i=0}^{n-1} (x_i - y_{(i+s) \bmod n} + c)^2 \\
&= \sum_{i=0}^{n-1} (x_i^2 + y_{(i+s) \bmod n}^2 + 2cx_i - 2cy_{(i+s) \bmod n} + c^2) - 2 \sum_{i=0}^{n-1} x_i y_{(i+s) \bmod n} \\
&= \sum_{i=0}^{n-1} (x_i^2 + y_i^2 + 2cx_i - 2cy_i + c^2) - 2 \sum_{i=0}^{n-1} x_i y_{(i+s) \bmod n} \\
&= \left[\sum_{i=0}^{n-1} (x_i^2 + y_i^2) + 2c \sum_{i=0}^{n-1} (x_i - y_i) + nc^2 \right] - 2 \sum_{i=0}^{n-1} x_i y_{(i+s) \bmod n}.
\end{aligned}$$

The first term depends solely on the inputs and the variable c , while the second term depends solely on the inputs and the variable s . Thus the two terms can be optimized separately. The first term can be optimized in $O(n)$ time by solving for when the derivative, which is linear in c , is zero. The second term can be computed, for each $s \in \{0, 1, \dots, n-1\}$, in $O(n \lg n)$ time using $(+, \cdot)$ convolution (and therefore optimized in the same time). Specifically, define the vectors

$$\begin{aligned}\vec{x}' &= \langle x_0, x_1, \dots, x_{n-1}; \underbrace{0, 0, \dots, 0}_n \rangle, \\ \vec{y}' &= \langle y_{n-1}, y_{n-2}, \dots, y_0; y_{n-1}, y_{n-2}, \dots, y_0 \rangle.\end{aligned}$$

Then, for $s' \in \{0, 1, \dots, n-1\}$, the $(n + s')$ th entry of the convolution $\vec{x}' * \vec{y}'$ is

$$\sum_{i=0}^{n+s'} x'_i y'_{n+s'-i} = \sum_{i=0}^{n-1} x_i y_{(i-s'-1) \bmod n},$$

which is the desired entry if we let $s' = n - 1 - s$. We can compute the entire convolution in $O(n \lg n)$ time using the Fast Fourier Transform. \square

The above result can be generalized to ℓ_p for any fixed *even* integer p . When $p \geq 4$, expanding the objective and rearranging the terms results in

$$\begin{aligned}\sum_{i=0}^{n-1} (x_i - y_{(i+s) \bmod n} + c)^p &= \sum_{i=0}^{n-1} \sum_{j=0}^p \binom{p}{j} (x_i - y_{(i+s) \bmod n})^{p-j} c^j \\ &= \sum_{j=0}^p \left(\binom{p}{j} \sum_{i=0}^{n-1} (x_i - y_{(i+s) \bmod n})^{p-j} \right) c^j,\end{aligned}$$

which is a degree- p polynomial in c , all of whose coefficients can be computed for all values of s by computing $O(p^2)$ convolutions.

Theorem 4. *The ℓ_p necklace alignment problem with p even can be solved in $O(p^2 n \lg n)$ time on a real RAM.*

4 ℓ_∞ Necklace Alignment and $(\min, +)$ Convolution

4.1 Reducing ℓ_∞ Necklace Alignment to $(\min, +)$ Convolution

First we show the relation between ℓ_∞ necklace alignment and $(\min, +)$ convolution. We need the following basic fact:

Fact 5. *For any vector $\vec{z} = \langle z_0, z_1, \dots, z_{n-1} \rangle$ and $c = -\frac{1}{2} (\min_{i=0}^{n-1} z_i + \max_{i=0}^{n-1} z_i)$, the minimum value of $\max_{i=0}^{n-1} |z_i + c|$ is*

$$\frac{1}{2} \left(\max_{i=0}^{n-1} z_i - \min_{i=0}^{n-1} z_i \right).$$

Instead of using $(\min, +)$ convolution directly, we use two equivalent forms, $(\min, -)$ and $(\max, -)$ convolution:

Theorem 6. *The ℓ_∞ necklace alignment problem can be reduced in $O(n)$ time to one $(\min, -)$ convolution and one $(\max, -)$ convolution.*

Proof. For two necklaces \vec{x} and \vec{y} , we apply the $(\min, -)$ convolution to the following vectors:

$$\begin{aligned}\vec{x}' &= \langle x_0, x_1, \dots, x_{n-1}; \underbrace{\infty, \infty, \dots, \infty}_n \rangle, \\ \vec{y}' &= \langle y_{n-1}, y_{n-2}, \dots, y_0; y_{n-1}, y_{n-2}, \dots, y_0 \rangle.\end{aligned}$$

Then, for $s' \in \{0, 1, \dots, n-1\}$, the $(n+s')$ th entry of $\vec{x}' \underset{\min}{*} \vec{y}'$ is

$$\min_{i=0}^{n+s'} (x'_i - y'_{n+s'-i}) = \min_{i=0}^{n-1} (x_i - y_{(i-s') \bmod n}),$$

which is $\min_{i=0}^{n-1} (x_i - y_{(i+s) \bmod n})$ if we let $s' = n-1-s$. By symmetry, we can compute the $(\max, -)$ convolution $\vec{x}'' \underset{\max}{*} \vec{y}'$, where \vec{x}'' has $-\infty$'s in place of ∞ 's, and use it to compute $\max_{i=0}^{n-1} (x_i - y_{(i+s) \bmod n})$ for each $s \in \{0, 1, \dots, n-1\}$. Applying Fact 5, we can therefore minimize $\max_{i=0}^{n-1} |x_i - y_{(i+s) \bmod n} + c|$ over c , for each $s \in \{0, 1, \dots, n-1\}$. By brute force, we can minimize over s as well using $O(n)$ additional comparisons and time. \square

4.2 $(\min, -)$ Convolution in Nonuniform Linear Decision Tree

For our nonuniform linear decision tree results, we use the main theorem of Fredman's work on sorting $X + Y$:

Theorem 7. [27] *For any fixed set Γ of permutations of N elements, there is a comparison tree of depth $O(N + \lg |\Gamma|)$ that sorts any sequence whose rank permutation belongs to Γ .*

Theorem 8. *The $(\min, -)$ convolution of two vectors of length n can be computed in $O(n\sqrt{n})$ time in the nonuniform linear decision tree model.*

Proof. Let \vec{x} and \vec{y} denote the two vectors of length n , and let $\vec{x} \underset{\min}{*} \vec{y}$ denote their $(\min, -)$ convolution, whose k th entry is $\min_{i=0}^k (x_i - y_{k-i})$.

First we sort the set $D = \{x_i - x_j, y_i - y_j : |i - j| \leq d\}$ of pairwise differences between nearby x_i 's and nearby y_i 's, where $d \leq n$ is a value to be determined later. This set D has $N = O(nd)$ elements. The possible sorted orders of D correspond to cells in the arrangement of hyperplanes in \mathbb{R}^{2n} induced by all $\binom{N}{2}$ possible comparisons between elements in the set, and this hyperplane arrangement has $O(N^{4n})$ cells. By Theorem 7, there is a comparison tree sorting D of depth $O(N + n \lg N) = O(nd + n \lg n)$.

The comparisons we make to sort D enable us to compare $x_i - y_{k-i}$ versus $x_j - y_{k-j}$ for free, provided $|i - j| \leq d$, because $x_i - y_{k-i} < x_j - y_{k-j}$ precisely if $x_i - x_j < y_{k-i} - y_{k-j}$. Thus, in particular, we can compute

$$M_k(\lambda) = \min \left\{ x_i - y_{k-i} \mid i = \lambda, \lambda + 1, \dots, \min\{\lambda + d, n\} - 1 \right\}$$

for free (using the outcomes of the comparisons we have already made).

We can rewrite the k th entry $\min_{i=0}^k (x_i - y_{k-i})$ of $\vec{x} \underset{\min}{*} \vec{y}$ as $\min\{M_k(0), M_k(d), M_k(2d), \dots, M_k(\lceil k/d \rceil d)\}$, and thus we can compute it in $O(k/d) = O(n/d)$ comparisons between differences. Therefore all n entries can be computed in $O(nd + n \lg n + n^2/d)$ total time.

This asymptotic running time is minimized when $nd = \Theta(n^2/d)$, i.e., when $d^2 = \Theta(n)$. Substituting $d = \sqrt{n}$, we obtain a running time of $O(n\sqrt{n})$ in the nonuniform linear decision tree model. \square

Combining Theorems 6 and 8, we obtain the following result:

Corollary 9. *The ℓ_∞ necklace alignment problem can be solved in $O(n\sqrt{n})$ time in the nonuniform linear decision tree model.*

4.3 $(\min, -)$ Convolution in Real RAM via Geometric Dominance

Our algorithm on the real RAM uses the following geometric lemma from Chan's work on all-pairs shortest paths:

Lemma 10. [9, Lemma 2.1] *Given n points p_1, p_2, \dots, p_n in d dimensions, each colored either red or blue, we can find the P pairs (p_i, p_j) for which p_i is red, p_j is blue, and p_i dominates p_j (i.e., for all k , the k th coordinate of p_i is at least the k th coordinate of p_j), in $2^{O(d)}n^{1+\varepsilon} + O(P)$ time for arbitrarily small $\varepsilon > 0$.*

Theorem 11. *The $(\min, -)$ convolution of two vectors of length n can be computed in $O(n^2/\lg n)$ time on a real RAM.*

Proof. Let \vec{x} and \vec{y} denote the two vectors of length n , and let $\vec{x} \underset{\max}{*} \vec{y}$ denote their $(\max, -)$ convolution. (Symmetrically, we can compute the $(\min, -)$ convolution.) for each $i \in \{0, d, 2d, \dots, \lfloor n/d \rfloor d\}$, and for each $j \in \{0, 1, \dots, n-1\}$, we define the d -dimensional points

$$\begin{aligned} p_{\delta,i} &= (x_{i+\delta} - x_i, x_{i+\delta} - x_{i+1}, \dots, x_{i+\delta} - x_{i+d-1}), \\ q_{\delta,j} &= (y_{j-\delta} - y_i, y_{j-\delta} - y_{i-1}, \dots, y_{j-\delta} - y_{j-d-1}). \end{aligned}$$

(To handle boundary cases, define $x_i = \infty$ and $y_j = -\infty$ for indices i, j outside $[0, n-1]$.) For each $\delta \in \{0, 1, \dots, d-1\}$, we apply Lemma 10 to the set of red points $\{p_{\delta,i} : i = 0, d, 2d, \dots, \lfloor n/d \rfloor d\}$ and the set of blue points $\{q_{\delta,j} : j = 0, 1, \dots, n-1\}$, to obtain all dominating pairs $(p_{\delta,i}, q_{\delta,j})$.

Point $p_{\delta,i}$ dominates $q_{\delta,j}$ precisely if $x_{i+\delta} - x_{i+\delta'} \geq y_{j-\delta} - y_{j-\delta'}$ for all $\delta' \in \{0, 1, \dots, d-1\}$ (ignoring the indices outside $[0, n-1]$). By re-arranging terms, this condition is equivalent to $x_{i+\delta} - y_{j-\delta} \geq x_{i+\delta'} - y_{j-\delta'}$ for all $\delta' \in \{0, 1, \dots, d-1\}$. If we substitute $j = k - i$, we obtain that $(p_{\delta,i}, q_{\delta,k-i})$ is a dominating pair precisely if $x_{i+\delta} - y_{k-i-\delta} = \max_{\delta'=1}^{d-1} (x_{i+\delta'} - y_{k-i-\delta'})$. Thus, the set of dominating pairs gives us the maximum $M_k(i) = \max\{x_i - y_{k-i}, x_{i+1} - y_{k-i+1}, \dots, x_{\min\{i+d,n\}-1} - y_{\min\{k-i+d,n\}-1}\}$ for each i divisible by d and for each k . Also, there can be at most $O(n^2/d)$ such pairs for all i, j, δ , because there are $O(n/d)$ choices for i and $O(n)$ choices for j , and if $(p_{\delta,i}, q_{\delta,j})$ is a dominating pair, then $(p_{\delta',i}, q_{\delta',j})$ cannot be a dominating pair for any $\delta' \neq \delta$. (Here we assume that the max is achieved uniquely, which can be arranged by standard perturbation techniques or by breaking ties consistently [9].) Hence, the running time of the d executions of Lemma 10 is $d2^{O(d)}n^{1+\varepsilon} + O(n^2/d)$ time, which is $O(n^2/\lg n)$ if we choose $d = \alpha \lg n$ for a sufficiently small constant $\alpha > 0$. We can rewrite the k th entry $\max_{i=0}^k (x_i - y_{k-i})$ of $\vec{x} \underset{\max}{*} \vec{y}$ as $\max\{M_k(0), M_k(d), M_k(2d), \dots, M_k(\lfloor k/d \rfloor d)\}$, and thus we can compute it in $O(k/d) = O(n/d)$ time. Therefore all n entries can be computed in $O(n^2/d) = O(n^2/\lg n)$ time on a real RAM. \square

Combining Theorems 6 and 11, we obtain the following result:

Corollary 12. *The ℓ_∞ necklace alignment problem can be solved in $O(n^2/\lg n)$ time on a real RAM.*

Although we will present a slightly faster algorithm for $(\min, -)$ convolution in the next subsection, the approach described above will be useful later when we discuss the $(\text{median}, -)$ convolution problem.

4.4 $(\min, -)$ Convolution via Matrix Multiplication

Our next algorithm uses Chan's $O(n^3(\lg \lg n)^3 / \lg^2 n)$ algorithm for computing the $(\min, +)$ matrix multiplication of two $n \times n$ matrices [10] (to which all-pairs shortest paths also reduces). We establish a reduction from convolution to matrix multiplication.

Theorem 13. *If we can compute the $(\min, -)$ matrix multiplication of two $n \times n$ matrices in $T(n)$ time, then we can compute the $(\min, -)$ convolution of two vectors of length n in $O((n + T(\sqrt{n}))\sqrt{n})$ time.*

Proof. We claim that computing the $(\min, -)$ convolution $\vec{z} = \vec{x} \underset{\min}{*} \vec{y}$ reduces to the following $(\min, -)$ matrix multiplication:

$$P = \sqrt{n} \left\{ \underbrace{\begin{pmatrix} x_0 & x_1 & \cdots & x_{\sqrt{n}-1} \\ x_{\sqrt{n}} & x_{\sqrt{n}+1} & \cdots & x_{2\sqrt{n}-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-\sqrt{n}} & x_{n-\sqrt{n}+1} & \cdots & x_{n-1} \end{pmatrix}}_{\sqrt{n}} \underset{\min}{\cdot} \underbrace{\begin{pmatrix} y_{\sqrt{n}-1} & y_{\sqrt{n}} & \cdots & y_{n-2} & y_{n-1} \\ y_{\sqrt{n}-2} & y_{\sqrt{n}-1} & \cdots & y_{n-3} & y_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ y_1 & y_2 & \cdots & y_{n-\sqrt{n}-2} & y_{n-\sqrt{n}-1} \\ y_0 & y_1 & \cdots & y_{n-\sqrt{n}-1} & y_{n-\sqrt{n}} \end{pmatrix}}_{n-\sqrt{n}+1} \right\} \sqrt{n}.$$

The (i, j) th entry $p_{i,j}$ of this product P is

$$p_{i,j} = \min_{m=0}^{\sqrt{n}-1} (x_{i\sqrt{n}+m} - y_{j+\sqrt{n}-1-m}).$$

Let $\bar{k} = \lfloor k/\sqrt{n} \rfloor \sqrt{n}$ denote the next smaller multiple of \sqrt{n} from k . Now, given the product P above, we can compute each element z_k of the convolution \vec{z} as follows:

$$z_k = \min \left\{ \begin{array}{l} p_{0,k+1-\sqrt{n}}, p_{1,k+1-2\sqrt{n}}, p_{2,k+1-3\sqrt{n}}, \dots, p_{\lfloor k/\sqrt{n} \rfloor - 1, k - \lfloor k/\sqrt{n} \rfloor \sqrt{n}}, \\ x_{\bar{k}} - y_{k-\bar{k}}, x_{\bar{k}+1} - y_{k-\bar{k}-1}, \dots, x_k - y_0 \end{array} \right\}.$$

This min has $O(\sqrt{n})$ terms, and thus z_k can be computed in $O(\sqrt{n})$ time. The entire vector \vec{z} can therefore be computed in $O(n\sqrt{n})$ time, given the matrix product P .

It remains to show how to compute the rectangular product P efficiently, given an efficient square-matrix $(\min, -)$ multiplication algorithm. We simply break the product P into at most \sqrt{n} products of $\sqrt{n} \times \sqrt{n}$ matrices: the left term is the entire left matrix, and the right term is a block submatrix. The number of blocks is $\lceil (n - \sqrt{n} + 1)/\sqrt{n} \rceil \leq \sqrt{n}$. Thus the running time for the product is $O(T(\sqrt{n})\sqrt{n})$.

Summing the reduction cost and the product cost, we obtain a total cost of $O((n + T(\sqrt{n}))\sqrt{n})$. \square

Plugging in $T(n) = O(n^3 / \lg n)$ from [9] allows us to obtain an alternative proof of Theorem 11. Plugging in $T(n) = O(n^3(\lg \lg n)^3 / \lg^2 n)$ from [10] immediately gives us the following improved result:

Corollary 14. *The $(\min, -)$ convolution of two vectors of length n can be computed in $O(n^2(\lg \lg n)^3 / \lg^2 n)$ time on a real RAM.*

Combining Theorem 6 and Corollary 14, we obtain the following result:

Corollary 15. *The ℓ_∞ necklace alignment problem can be solved in $O(n^2(\lg \lg n)^3 / \lg^2 n)$ time on a real RAM.*

We remark that by the reduction in Theorem 13, any nontrivial lower bound for $(\min, -)$ convolution would imply a lower bound for $(\min, -)$ matrix multiplication and the all-pairs shortest path problem.

5 ℓ_1 Necklace Alignment and $(\text{median}, +)$ Convolution

5.1 Reducing ℓ_1 Necklace Alignment to $(\text{median}, +)$ Convolution

First we show the relation between ℓ_1 necklace alignment and $(\text{median}, +)$ convolution. We need the following basic fact [2]:

Fact 16. *For any vector $\vec{z} = \langle z_0, z_1, \dots, z_{n-1} \rangle$, $\sum_{i=0}^{n-1} |z_i + c|$ is minimized when $c = -\text{median}_{i=0}^{n-1} z_i$.*

Instead of using $(\text{median}, +)$ convolution directly, we use the equivalent form, $(\text{median}, -)$ convolution:

Theorem 17. *The ℓ_1 necklace alignment problem can be reduced in $O(n)$ time to one $(\text{median}, -)$ convolution.*

Proof. For two necklaces \vec{x} and \vec{y} , we apply the $(\text{median}, -)$ convolution to the following vectors, as in the proof of Theorem 6:

$$\begin{aligned}\vec{x}' &= \langle x_0, x_0, x_1, x_1, \dots, x_{n-1}, x_{n-1}; \underbrace{\infty, -\infty, \infty, -\infty, \dots, \infty, -\infty}_{2n} \rangle, \\ \vec{y}' &= \langle y_{n-1}, y_{n-1}, y_{n-2}, y_{n-2}, \dots, y_0, y_0; y_{n-1}, y_{n-1}, y_{n-2}, y_{n-2}, \dots, y_0, y_0 \rangle.\end{aligned}$$

$\vec{x}' \underset{\text{med}}{*} \vec{y}'$ is

$$\text{median}_{i=0}^{2(n+s')+1} (x'_i - y'_{2(n+s')+1-i}) = \text{median}_{i=0}^{n-1} (x_i - y_{(i-s'-1) \bmod n}),$$

which is $\text{median}_{i=0}^{n-1} (x_i - y_{(i+s) \bmod n})$ if we let $s' = n - 1 - s$. Applying Fact 16, we can therefore minimize $\text{median}_{i=0}^{n-1} |x_i - y_{(i+s) \bmod n} + c|$ over c , for each $s \in \{0, 1, \dots, n-1\}$. By brute force, we can minimize over s as well using $O(n)$ additional comparisons and time. \square

Our results for $(\text{median}, -)$ convolution use the following result of Frederickson and Johnson:

Theorem 18. [26] *The median element of the union of k sorted lists, each of length n , can be computed in $O(k \lg n)$ time and comparisons.*

5.2 $(\text{median}, -)$ Convolution in Nonuniform Linear Decision Tree

We begin with our results for the nonuniform linear decision tree model:

Theorem 19. *The $(\text{median}, -)$ convolution of two vectors of length n can be computed in $O(n\sqrt{n \lg n})$ time in the nonuniform linear decision tree model.*

Proof. As in the proof of Theorem 9, we sort the set $D = \{x_i - x_j, y_i - y_j : |i - j| \leq d\}$ of pairwise differences between nearby x_i 's and nearby y_i 's, where $d \leq n$ is a value to be determined later. By Theorem 7, this step requires $O(nd + n \lg n)$ comparisons between differences. These comparisons enable us to compare $x_i - y_{k-i}$ versus $x_j - y_{k-j}$ for free, provided $|i - j| \leq d$, because $x_i - y_{k-i} < x_j - y_{k-j}$ precisely if $x_i - x_j < y_{k-i} - y_{k-j}$. In particular, we can sort each list

$$L_k(\lambda) = \left\langle x_i - y_{k-i} \mid i = \lambda, \lambda + 1, \dots, \min\{\lambda + d, n\} - 1 \right\rangle$$

for free. By Theorem 18, we can compute the median of $L_k(0) \cup L_k(d) \cup L_k(2d) \cup \dots \cup L_k(\lceil k/d \rceil d)$, i.e., $\text{median}_{i=0}^k(x_i - y_{k-i})$, in $O((k/d) \lg d) = O((n/d) \lg d)$ comparisons. Also, in the same asymptotic number of comparisons, we can binary search to find where the median fits in each of the $L_k(\lambda)$ lists, and therefore which differences are smaller and which differences are larger than the median. This median is the k th entry of $\vec{x} \underset{\text{med}}{*} \vec{y}$. Therefore, we can compute all n entries of $\vec{x} \underset{\text{med}}{*} \vec{y}$ in $O(nd + n \lg n + (n^2/d) \lg d)$ comparisons. This asymptotic running time is minimized when $nd = \Theta((n^2/d) \lg d)$, i.e., when $d^2 / \lg d = \Theta(n)$. Substituting $d = \sqrt{n \lg n}$, we obtain a running time of $O(n\sqrt{n \lg n})$ in the nonuniform linear decision tree model. \square

Combining Theorems 17 and 19, we obtain the following result:

Corollary 20. *The ℓ_1 necklace alignment problem can be solved in $O(n\sqrt{n \lg n})$ time in the nonuniform linear decision tree model.*

5.3 $(\min, -)$ Convolution in Real RAM via Geometric Dominance

Now we turn to the analogous results for the real RAM:

Theorem 21. *The $(\text{median}, -)$ convolution of two vectors of length n can be computed in $O(n^2(\lg \lg n)^2 / \lg n)$ time on a real RAM.*

Proof. Let \vec{x} and \vec{y} denote the two vectors of length n , and let $\vec{x} \underset{\text{med}}{*} \vec{y}$ denote their $(\text{median}, -)$ convolution. For each permutation π on the set $\{0, 1, \dots, d-1\}$, for each $i \in \{0, d, 2d, \dots, \lfloor n/d \rfloor d\}$, and for each $j \in \{0, 1, \dots, n-1\}$, we define the $(d-1)$ -dimensional points

$$\begin{aligned} p_{\pi,i} &= (x_{i+\pi(0)} - x_{i+\pi(1)}, x_{i+\pi(1)} - x_{i+\pi(2)}, \dots, x_{i+\pi(d-2)} - x_{i+\pi(d-1)}), \\ q_{\pi,j} &= (y_{j-\pi(0)} - y_{j-\pi(1)}, y_{j-\pi(1)} - y_{j-\pi(2)}, \dots, y_{j-\pi(d-2)} - y_{j-\pi(d-1)}), \end{aligned}$$

(To handle boundary cases, define $x_i = \infty$ and $y_j = -\infty$ for indices i, j outside $[0, n-1]$.) For each permutation π , we apply Lemma 10 to the set of red points $\{p_{\pi,i} : i = 0, d, 2d, \dots, \lfloor n/d \rfloor d\}$ and the set of blue points $\{q_{\pi,j} : j = 0, 1, \dots, n-1\}$, to obtain all dominating pairs $(p_{\pi,i}, q_{\pi,j})$.

Point $p_{\pi,i}$ dominates $q_{\pi,j}$ precisely if $x_{i+\pi(\delta)} - x_{i+\pi(\delta+1)} \geq y_{j-\pi(\delta)} - y_{j-\pi(\delta+1)}$ for all $\delta \in \{0, 1, \dots, d-2\}$ (ignoring the indices outside $[0, n-1]$). By re-arranging terms, this condition is equivalent to $x_{i+\pi(\delta)} - y_{j-\pi(\delta)} \geq x_{i+\pi(\delta+1)} - y_{j-\pi(\delta+1)}$ for all $\delta \in \{0, 1, \dots, d-2\}$, i.e., π is a sorting permutation of $\langle x_i - y_j, x_{i+1} - y_{j-1}, \dots, x_{i+d-1} - y_{j-d+1} \rangle$. If we substitute $j = k - i$, we obtain that $(p_{\pi,i}, q_{\pi,k-i})$ is a dominating pair precisely if π is a sorting permutation of the list $L_k(i) = \langle x_i - y_{k-i}, x_{i+1} - y_{k-i+1}, \dots, x_{\min\{i+d, n\}-1} - y_{\min\{k-i+d, n\}-1} \rangle$. Thus, the set of dominating pairs gives us the sorted order of $L_k(i)$ for each i divisible by d and for each k . Also, there can be at most $O(n^2/d)$ total dominating pairs $(p_{\pi,i}, q_{\pi,j})$ over all i, j, π , because there are $O(n/d)$ choices for i and $O(n)$ choices for j , and if $(p_{\pi,i}, q_{\pi,j})$ is a dominating pair, then $(p_{\pi',i}, q_{\pi',j})$ cannot be a dominating pair for any permutation $\pi' \neq \pi$. (Here we assume that the sorted order

is unique, which can be arranged by standard perturbation techniques or by breaking ties consistently [9].) Hence, the running time of the $d!$ executions of Lemma 10 is $d! 2^{O(d)} n^{1+\varepsilon} + O(n^2/d)$ time, which is $O(n^2 \lg \lg n / \lg n)$ if we choose $d = \alpha \lg n / \lg \lg n$ for a sufficiently small constant $\alpha > 0$. By Theorem 18, we can compute the median of $L_k(0) \cup L_k(d) \cup L_k(2d) \cup \dots \cup L_k(\lceil k/d \rceil d)$, i.e., $\text{median}_{i=0}^k(x_i - y_{k-i})$, in $O((k/d) \lg d) = O((n/d) \lg d)$ comparisons. Also, in the same asymptotic number of comparisons, we can binary search to find where the median fits in each of the $L_k(\lambda)$ lists, and therefore which differences are smaller and which differences are larger than the median. This median is the k th entry of $\vec{x} \underset{\text{med}}{*} \vec{y}$. Therefore all n entries can be computed in $O(n^2(\lg d)/d) = O(n^2(\lg \lg n)^2 / \lg n)$ time on a real RAM. \square

Combining Theorems 17 and 21, we obtain the following result:

Corollary 22. *The ℓ_1 necklace alignment problem can be solved in $O(n^2(\lg \lg n)^2 / \lg n)$ time on a real RAM.*

As before, this approach likely cannot be improved beyond $O(n^2 / \lg n)$, because such an improvement would require an improvement to Lemma 10, which would in turn improve the fastest known algorithm for all-pairs shortest paths in dense graphs [10].

In contrast to (median, +) convolution, (mean, +) convolution is trivial to compute in linear time by inverting the two summations.

6 Conclusion

The convolution problems we consider here have connections to many classic problems, and it would be interesting to explore whether the structural information extracted by our algorithms could be used to devise faster algorithms for these classic problems. For example, does the antidiagonal information of the $X + Y$ matrix lead to a $o(n^2 \lg n)$ -time algorithm for sorting $X + Y$? We believe that any further improvements to our convolution algorithms would require progress and/or have interesting implications on all-pairs shortest paths [9].

Our (min, -)-convolution algorithms give subquadratic algorithms for *polyhedral 3SUM*: given three lists, $A = \langle a_0, a_1, \dots, a_{n-1} \rangle$, $B = \langle b_0, b_1, \dots, b_{n-1} \rangle$, and $C = \langle c_0, c_1, \dots, c_{2n-2} \rangle$, such that $a_i + b_j \leq c_{i+j}$ for all $0 \leq i, j < n$, decide whether $a_i + b_j = c_{i+j}$ for any $0 \leq i, j < n$. This problem is a special case of 3SUM, and this special case has an $\Omega(n^2)$ lower bound in the 3-linear decision tree model [23]. Our results solve polyhedral 3SUM in $O(n^2 / \lg n)$ time in the 4-linear decision tree model, and in $O(n\sqrt{n})$ time in the nonuniform 4-linear decision tree model, solving an open problem of Erickson [21]. Can these algorithms be extended to solve 3SUM in subquadratic time in the (nonuniform) decision tree model?

Acknowledgments

This work was initiated at the 20th Bellairs Winter Workshop on Computational Geometry held January 28–February 4, 2005. We thank the other participants of that workshop—Greg Aloupis, Justin Colannino, Mirela Damian-Iordache, Vida Dujmović, Francisco Gomez-Martin, Danny Krizanc, Erin McLeish, Henk Meijer, Patrick Morin, Mark Overmars, Suneeta Ramaswami, David Rappaport, Diane Souvaine, Ileana Streinu, David Wood, Godfried Toussaint, Remco Velthkamp, and Sue Whitesides—for helpful discussions and contributing to a fun and creative atmosphere. We particularly thank the organizer, Godfried Toussaint, for posing the problem to us. The last author

would also like to thank Luc Devroye for pointing out the easy generalization of the ℓ_2 necklace alignment problem to ℓ_p for any fixed even integer p .

References

- [1] Aloupis, G., Fevens, T., Langerman, S., Matsui, T., Mesa, A., Nuñez, Y., Rappaport, D., Toussaint, G.: Algorithms for computing geometric measures of melodic similarity. *Journal of Computational Music* **30**(3), 67–76 (2006)
- [2] Ardila, Y.J.P., Clifford, R., Iliopoulos, C.S., Landau, G.M., Mohamed, M.: Necklace swap problem for rhythmic similarity measures. *International Journal of Computational Methods* **5**(3), 351–363 (2008)
- [3] Baran, I., Demaine, E.D., Pătraşcu, M.: Subquadratic algorithms for 3SUM. *Algorithmica* **50**(4), 584–596 (2008). Special issue of selected papers from the 9th Workshop on Algorithms and Data Structures, 2005
- [4] Bellman, R., Karush, W.: Mathematical programming and the maximum transform. *Journal of the Society for Industrial and Applied Mathematics* **10**(3), 550–567 (1962)
- [5] Bernstein, D.J.: Fast multiplication and its applications. In: J. Buhler, P. Stevenhagen (eds.) *Algorithmic Number Theory*, vol. 44. MSRI Publications (2008)
- [6] Blum, M., Floyd, R.W., Pratt, V., Rivest, R.L., Tarjan, R.E.: Time bounds for selection. *Journal of Computer and System Sciences* **7**(4), 448–461 (1973)
- [7] Bussieck, M., Hassler, H., Woeginger, G.J., Zimmermann, U.T.: Fast algorithms for the maximum convolution problem. *Operations Research Letters* **15**(3), 133–141 (1994)
- [8] Cardinal, J., Kremer, S., Langerman, S.: Juggling with pattern matching. *Theory of Computing Systems* **39**(3), 425–437 (2006)
- [9] Chan, T.M.: All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. *Algorithmica* **50**, 236–243 (2008)
- [10] Chan, T.M.: More algorithms for all-pairs shortest paths in weighted graphs. *SIAM Journal on Computing* **39**, 2075–2089 (2010)
- [11] Clifford, P., Clifford, R.: Self-normalised distance with don’t cares. In: B. Ma, K. Zhang (eds.) *Combinatorial Pattern Matching, Lecture Notes in Computer Science*, vol. 4580, pp. 63–70. Springer Berlin Heidelberg (2007)
- [12] Clifford, P., Clifford, R.: Simple deterministic wildcard matching. *Information Processing Letters* **101**(2), 53–54 (2007)
- [13] Clifford, P., Clifford, R., Iliopoulos, C.: Fourier transform methods for δ and (δ, γ) matching and other measures of string similarity. Tech. Rep. TR-04-09, King’s College London (2004)
- [14] Clifford, P., Clifford, R., Iliopoulos, C.: Faster algorithms for δ , γ -matching and related problems. In: *Combinatorial Pattern Matching, Lecture Notes in Computer Science*, vol. 3537, pp. 71–90. Springer Berlin Heidelberg (2005)

- [15] Cohn, H., Kleinberg, R., Szegedy, B., Umans, C.: Group-theoretic algorithms for matrix multiplication. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 379–388 (2005)
- [16] Colannino, J., Damian, M., Hurtado, F., Iacono, J., Meijer, H., Ramaswami, S., Toussaint, G.: An $O(n \log n)$ -time algorithm for the restriction scaffold assignment. *Journal of Computational Biology* **13**(4), 979–989 (2006)
- [17] Cole, R., Hariharan, R.: Verifying candidate matches in sparse and wildcard matching. In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing, pp. 592–601. ACM Press, Montréal, Canada (2002)
- [18] Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation* **19**, 297–301 (1965)
- [19] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press (2001)
- [20] Demaine, E.D., Mitchell, J.S.B., O’Rourke, J.: Problem 41: Sorting $X + Y$ (pairwise sums). In: The Open Problems Project (2006). <http://cs.smith.edu/~orourke/TOPP/P41.html>
- [21] Demaine, E.D., O’Rourke, J.: Open problems from CCCG 2005. In: Proceedings of the 18th Canadian Conference on Computational Geometry. Kingston, Canada (2006)
- [22] Díaz-Báñez, J.M., Farigu, G., Gómez, F., Rappaport, D., Toussaint, G.T.: El compás flamenco: A phylogenetic analysis. In: Proceedings of BRIDGES: Mathematical Connections in Art, Music, and Science, pp. 61–70. Winfield, Kansas (2004)
- [23] Erickson, J.: Lower bounds for linear satisfiability problems. *Chicago Journal of Theoretical Computer Science* **1999**(8) (1999)
- [24] Felzenszwalb, P.F., Huttenlocher, D.P.: Distance transforms of sampled functions. Tech. Rep. TR2004-1963, Faculty of Computing and Information Science, Cornell University (2004)
- [25] Fischer, M.J., Paterson, M.S.: String-matching and other products. In: Complexity of computation, *SIAM-AMS Proceedings*, vol. VII, pp. 113–125. American Mathematical Society (1974). Proceedings of the SIAM-AMS Applied Mathematics Symposium, New York, 1973.
- [26] Frederickson, G.N., Johnson, D.B.: The complexity of selection and ranking in $X + Y$ and matrices with sorted columns. *Journal of Computer and System Sciences* **24**(2), 197–208 (1982)
- [27] Fredman, M.L.: How good is the information theory bound in sorting? *Theoretical Computer Science* **1**(4), 355–361 (1976)
- [28] Fredman, M.L.: New bounds on the complexity of the shortest path problem. *SIAM Journal on Computing* **5**(1), 83–89 (1976)
- [29] Gauss, C.F.: Werke, vol. 3. Königlichen Gesellschaft der Wissenschaften, Göttingen (1866)
- [30] Heideman, M.T., Johnson, D.H., Burrus, C.S.: Gauss and the history of the fast Fourier transform. *Archive for History of Exact Sciences* **34**(3), 265–277 (1985)

- [31] Indyk, P.: Faster algorithms for string matching problems: Matching the convolution bound. In: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, pp. 166–173. Palo Alto, California (1998)
- [32] Kalai, A.: Efficient pattern-matching with don’t cares. In: Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 655–656. San Francisco, California (2002)
- [33] Maragos, P.: Differential morphology. In: S. Mitra, G. Sicuranza (eds.) Nonlinear Image Processing, chap. 10, pp. 289–329. Academic Press (2000)
- [34] Moreau, J.J.: Inf-convolution, sous-additivité, convexité des fonctions numériques. Journal de Mathématiques Pures et Appliquées, Neuvième Série **49**, 109–154 (1970)
- [35] Rockafellar, R.T.: Convex Analysis. Princeton Mathematical Series. Princeton University Press (1970)
- [36] Schönhage, A., Paterson, M., Pippenger, N.: Finding the median. Journal of Computer and System Sciences **13**(2), 184–199 (1976)
- [37] Steiger, W.L., Streinu, I.: A pseudo-algorithmic separation of lines from pseudo-lines. Information Processing Letters **53**(5), 295–299 (1995)
- [38] Strömberg, T.: The operation of infimal convolution. Dissertationes Mathematicae (Rozprawy Matematyczne) **352**, 58 (1996)
- [39] Toussaint, G.: The geometry of musical rhythm. In: Revised Papers from the Japan Conference on Discrete and Computational Geometry, *Lecture Notes in Computer Science*, vol. 3742, pp. 198–212. Tokyo, Japan (2004)
- [40] Toussaint, G.T.: A comparison of rhythmic similarity measures. In: Proceedings of the 5th International Conference on Music Information Retrieval, pp. 242–245. Barcelona, Spain (2004). A longer version appears as Technical Report SOCS-TR-2004.6, School of Computer Science, McGill University, August 2004.